Cédric Meichtry
Cédric Gaudard

Projet Client IRC:

Rapport

Introduction

Dans le cadre du cours de réseaux informatiques en 2ème année de système de communication à l'epfl, nous avons programmé un Client IRC en java. Ce projet a pour objectif, après avoir compris les concepts régissant les transferts de données sur un réseau, de développer un code Client Chat, conforme aux RFC remplissant les critères suivants: Connexion au serveur, joindre plusieurs canaux, envoyer et recevoir des messages privés, sortir d'un canal et quitter proprement le serveur.

Description du programme

Concept

Après avoir étudier le cahier des charges, nous avons décidé de structurer notre programme en deux classes principales:

La première, Chat va créer et gérer la majeure partie de l'interface graphique, établir et entretenir la connexion avec le serveur au niveau des sockets. De plus elle traite aussi toutes les interventions de l'utilisateur et transmet ces messages au serveur en respectant la RFC 2812.

La seconde et la classe Traitement qui va être sollicitée sur appel de la classe Chat lorsqu'il s'agira de décoder un message venant du serveur pour le rendre visible à l'utilisateur. Elle découpe le message et réagit en fonction des commandes et des paramètres, elle met à jour l'interface graphique et redonne à la classe Chat la réponse au message synchrone qui requiert une réponse automatique.

Application au code

La Méthode Chat, contient la méthode d'exécution main pour lancer le programme (Cette méthode aurait pu être mise dans une classe séparée); elle va aller tout d'abord chercher l'adresse IP du serveur chat, puis elle construit un objet Chat;

Le constructeur de Chat est responsable de construire l'interface graphique et de créer la connexion par l'ouverture d'un socket et des tampons adéquats.

Ensuite, la méthode main lance login() qui va cette fois établir le dialogue avec le serveur et passer les premiers paramètres nécessaires à une identification et/ou un enregistrement (commande PASS, NICK, USER)

Cédric Meichtry
Cédric Gaudard

La dernière étape est la création d'un processus et son lancement. A partir de là, c'est la méthode run() qui prend le relais pour l'exécution de la suite des opérations.

Mais finissons de décrire les autres tâches de la classe Chat. Toutes les interventions de l'utilisateur sont récupérées et traitées par cette classe; que cela soit pour l'envoi de message ou pour joindre un nouveau canal, lister des noms d'utilisateurs ou autres. Lors d'un événements de ce types, la classe envoi au tampon le message à transmettre, que le tampon écrit dans le socket et qui sera vraisemblablement (si le réseau n'a pas de problèmes graves) délivré au serveur IRC.

Revenons à la méthode run() qui lit en permanence le tampon pour y détecter des nouvelles transmissions. Lors de la réception d'un message venant du serveur le processus va réagir. Il lit le message et l'envoi à la classe Traitement pour que le message soit analysé et affiché dans l'interface graphique. De plus si le message requiert une réponse automatique la classe traitement va la fournir et la méthode run() va le transmettre.

Voyons d'un peu plus près le fonctionnement de la classe Traitement:
Le constructeur fait appel à deux méthodes:
Segmentation() se charge de découper le message et de l'enregistrer dans les différentes variables telles que command (reprenant la commande envoyée), params[] (un tableau de paramètres), prefix (définissant l'auteur du message) etc.
Command() quant à elle se charge uniquement du triage. Elle va rediriger l'exécution vers la bonne méthode en fonction du numéro ou du texte de la commande.
Après il y a une méthode par commande différente qui fait afficher au bon endroit les différents paramètres pour permettre une lecture plus agréable à l'utilisateur des messages reçus.

A chaque niveau de traitement les méthodes détectent s'il y a une erreur dans le message reçu ou si la commande n'est pas implémentée et l'affiche.

Il reste les deux interfaces TextAreaBig et TextAreaSmall, elles servent uniquement à ce que ces objets soient visibles depuis les deux classes et permette l'écriture dans l'interface depuis les deux endroits.

Mode d'emploi

On lance le programme et il se connecte au serveur inscrit dans le code.
- Pour demander la liste des canaux cliquez sur le bouton **liste.**
- Pour joindre un canal sélectionnez le dans la zone de texte gauche, puis cliquez sur le bouton **join.**
- Pour quitter un canal sélectionnez le dans la zone de texte gauche, puis cliquez sur le bouton **quit (ch).**
- Pour quittez le programme cliquez sur le bouton **quit.**

Les commandes utiles pour un chat minimum sont:
NICK lui /change mon nick en lui ou le crée s'il n'est pas connecté
PRIVMSG #essai :msg , msg est envoyé au canal essai en entier

PRIVMSG toto :salut toto tu vas bien?
//ici on envoie à toto uniquement attention à ne pas oublier les ":" pour que cela fonctionne
PART #essai /quitte le canal essai (si jamais vous avez une panne de bouton)
JOIN #essai /joint le canal essai (si jamais vous avez une panne de bouton)

Pour de plus amples informations référez vous à la rfc 2812

Les Traces

Nous avons testé notre client sur différents serveur, dont principalement les suivants: irc.voila.fr port 6667, irc.forestnet.org port 6667, nalova.dyndns.org port 6667 ainsi que sur les serveurs de test in1sun1 port 1980 et 1981 ou d'autres sur les différents postes in1sun??.

irc.forestnet.org et nalova.dyndns.org

Nous avons à l'aide de usnoop et usnoop –V observé le trafic qui s'effectue entre les différentes machines. Malheureusement n'ayant pas les droits root nous ne pouvons voir qu'un trafic entre notre machine et une autre de l'epfl le plus souvent dinfs, nous avons pu observer une fois la requête dns pour le numéro IP du serveur (fichier non disponible), sinon tout le reste est inutilisable. On voit juste du trafic.

Le serveur de test

Avec le serveur de test, on peut observer que les commandes fonctionnent et qu'il est possible de faire du chat avec notre programme sans erreurs.

Cédric Meichtry
Cédric Gaudard

Documentation des classes et des méthodes

Chat.java

Cette classe construit la GUI et l'interface réseau

Traitement.java

Cette classe effectue le traitement des messages reçus par le client et met à jour les variables extractées de la ligne de commande reçue puis, dans une deuxième phase met à jour l'interface graphique pour que l'utilisateur soit informé des derniers événements. Le constructeur est appelé pour chaque nouvelle donnée reçue et crée un objet.

TextAreaBig.java

interface représentant la grande zone de texte dans laquelle on écrira les messages reçus et émis. Cette zone de texte sera accessible depuis chat et traitement.

TextAreaSmall.java

interface représentant la petite zone de texte dans laquelle on écrira la liste des clients connectes au serveur. Cette zone de texte sera accessible depuis chat et traitement.

TextAreaSmallChannel.java

interface représentant la petite zone de texte dans laquelle on écrira la liste des channels du serveur auquel on est connecté. Cette zone de texte sera accessible depuis chat et traitement.

Cédric Meichtry
Cédric Gaudard

Table des matières